

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

**UTILITY PATENT APPLICATION TRANSMITTAL
UNDER 37 CFR 1.53(b)**

Address to: Assistant Commissioner for Patents Box Patent Application Washington, DC 20231	Attorney Docket No.	AM9-99-0209
	Inventor(s)	Choy
	Express Mail Label No.	EL640503360US
	Total Pages	34

Title of Application:

SYSTEM AND METHOD FOR ENSURING REFERENTIAL INTEGRITY FOR HETEROGENEOUSLY SCOPED REFERENCES IN AN INFORMATION MANAGEMENT SYSTEM

Transmitted with the patent application are the following:

- 1 Page(s) Transmittal form (plus one copy)
- 21 Page(s) Specification, claims, abstract
- 5 Page(s) Drawings
- 4 Page(s) Declaration and Power of Attorney
- 1 Page(s) Recordation Form Cover Sheet
- 2 Page(s) Assignment of the Invention to International Business Machines Corporation

This application is a: ☐ Continuation ☐ Divisional ☐ Continuation-in-Part of prior application Serial No. _____

Fee Calculation

	Claims Filed		Extra	Rate	Fees
Basic Fee					\$690.00
Total Claims	22	- 20 =	2	× \$18.00	36.00
Independent Claims	4	- 3 =	1	× \$78.00	78.00
Multiple Dependent Claim				+ \$260.00	
				Assignment	\$ 40.00
				TOTAL	\$844.00

The Commissioner is hereby authorized to charge fees of \$844 to Deposit Account 09-0441 as required under 37 CFR 1.16 or 1.17.

EXPRESS MAIL CERTIFICATE

I hereby certify that the above paper/fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated below and is addressed to the Assistant Commissioner for Patents, Washington, DC 20231

Date of Deposit: July 12, 2000
 Person mailing paper/fee: Jeanne Gahagan

Signature: Jeanne Gahagan

Respectfully submitted,

John L. Rogitz (#33,549)
 Attorney for Applicant(s)
 Telephone (619) 338-8075
 Rogitz & Associates
 750 B Street, Suite 3120
 San Diego, California 92101

SYSTEM AND METHOD FOR ENSURING REFERENTIAL INTEGRITY FOR HETEROGENEOUSLY SCOPED REFERENCES IN AN INFORMATION MANAGEMENT SYSTEM

1. Field of the Invention

The present invention relates to methods and systems for ensuring referential integrity ("no dangling pointers") in databases containing heterogeneously scoped references.

2. Background of the Invention

Information management systems (IMS) such as digital library servers typically use a relational database management system (RDBMS) to manage data records in a database. As an example, an IMS might manage document data, with the desire that the RDBMS maintain so-called "referential integrity" for elements in the database. What is meant by "referential integrity" is that if a row of a table (referred to as a "target" table) that is referenced by a row in another table (referred to as a "source" table) by means of a foreign key (referred to as a "pointer") is sought to be deleted or changed, either the deletion or change is disallowed or the pointer is removed from the source table, so that no "dangling pointers" (i.e., links to nonexistent database elements) remain.

This is but one example of what is meant by "referential integrity". In general, when one entity refers to another, a dependency relationship is created that represents a referential constraint the integrity of which might be desirable to maintain. As recognized herein, not all such constraints need have their integrity maintained; rather, deciding which constraints to maintain is a balance between application requirements and what can be maintained efficiently.

In existing RDBMS systems such as the system known as "DB2" and marketed by the present assignee, maintaining referential integrity assumes homogenous linking, i.e., that all links from a

column of a source table point to tuples in a single predetermined target table. As recognized herein, however, in some cases, such as, for example, digital libraries, a column of links can exist in a table that do not all point to a single common target table. This is because different data types can exist in these cases, so that, for example, one link in a column of a source table might point to a target table representing a particular type of document, another link in the column might point to a target table representing another particular type of document, while yet a third link might point to an image, all of them nonetheless being interrelated. Such links are referred to as "heterogeneously scoped" links, whereby the scope (or target) of these links consists of multiple pre-determined tables. A special case is "unscoped" links, whereby the target tuple of a link can be in any table.

Accordingly, "heterogenous scoping" refers to the ability to create and manage a column of links whose targets are not restricted to being tuples of a single table. It will readily be appreciated by the skilled artisan that heterogenous scoping would provide a flexible and generic linking ability that can be exploited by, e.g., a digital library system to support a relatively more complex data model.

One way to provide heterogenous linking referential integrity is to modify the RDBMS. This is not a trivial task. Accordingly, the present invention is directed to a system and method for providing heterogenous linking referential integrity without modifying the RDBMS, but rather by providing a transparent software layer within the IMS or between the RDBMS and IMS that exploits existing RDBMS functionality to extend referential integrity to heterogeneously scoped links.

SUMMARY OF THE INVENTION

5 The invention is a general purpose computer programmed according to the inventive steps herein. The invention can also be embodied as an article of manufacture - a machine component - that is used by a digital processing apparatus and which tangibly embodies a program of instructions that are executable by the digital processing apparatus to undertake the present invention. This invention is realized in a critical machine component that causes a digital processing apparatus to perform the inventive method steps herein. The invention is also a computer-implemented method for undertaking the acts disclosed below.

10 Accordingly, a system is disclosed for providing referential integrity for heterogenous links. The system includes an RDBMS that provides referential integrity for homogenous links, and an IMS communicates with the RDBMS. A software layer is within the IMS or between the RDBMS and IMS for causing the RDBMS to provide referential integrity for heterogenous links.

15 In a preferred embodiment, the software layer includes at least one table that is maintained by the software layer. The table is accessed upon an attempted deletion or updating of a link, and the attempted deletion or updating is selectively disallowed by the software layer, based on the table. If desired, the software layer can establish at least one trigger, such as an update trigger or delete trigger, that is useful in selectively disallowing operations. Also, if desired the preferred software layer can include at least one stored procedure accessible by an application to insert or delete a tuple while ensuring referential integrity in heterogenous links associated with the tuple.

20 In another aspect, a computer-implemented method for preventing dangling pointers in heterogeneously scoped links includes providing at least one heterogeneously scoped link (HSL) table that is separate from an RDBMS having stored therein plural tuples arranged in tables. At least one

table has a heterogeneously scoped link column, and the table is associated with the heterogeneously scoped link column. The method envisions accessing the HSL table to ensure referential integrity in the RDBMS.

In still another aspect, a computer program product includes computer usable code means that are programmed with logic for ensuring referential integrity in an RDBMS. The RDBMS has at least one table with at least one column of heterogeneously scoped links. The program product includes computer readable code means for accessing a table that is not part of the RDBMS, e.g., that is maintained by the program product. Computer readable code means use the table to ensure that operations on tuples do not result in a heterogeneously scoped link pointing to no tuple.

The details of the present invention, both as to its structure and operation, can best be understood in reference to the accompanying drawings, in which like reference numerals refer to like parts, and in which:

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of the present system;

Figure 2 is a flow chart of the logic for creating an RI table;

Figure 3 is a schematic representation of an RI table;

Figure 4 is a flow chart of the logic for using the RI table;

Figure 5 is a schematic representation of a user-defined type for a heterogeneously scoped or unscoped link;

Figure 6 is a schematic representation of a Link table;

Figure 7 is a schematic representation of a Scope table;

Figure 8 is a schematic representation of a Column table; and

Figure 9 is a flow chart of alternate logic for using the tables shown in Figures 6-8 to ensure referential integrity of heterogeneously scoped links.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

5 Referring initially to Figure 1, a system is shown, generally designated 10. To give the present invention context, an exemplary system 10 having an information management system (IMS) such as a digital library that cooperates with an underlying relational database management system (RDBMS) is described herein, it being understood that the present principles for ensuring referential integrity of unscoped or heterogeneously scoped links (sometimes collectively referred to herein for convenience as "heterogeneously scoped links", or simply "heterogenous links") applies to system architectures other than the one shown.

10 Accordingly, the system 10 includes at least one user computer 12 having a software-implemented application 14 that generates queries. Input to the user computer 12 is via one or more input devices 16, and query results can be output on an output device 18. The input device 16 can be any suitable device, such as a keyboard, keypad, mouse, joystick, trackball, voice-recognition software, and so on. The output device 18 can be a monitor, a speaker, another computer or computer network, a printer, and so on.

15 As shown in Figure 1, the user computer 12 communicates with an information management system (IMS) 20, such as a digital library server, via an IMS communication path 22 using high-level application programming interfaces (API). The IMS 20 communicates with a relational database system (RDBMS) 24, such as the present assignee's DB2, that stores records of documents managed

20

by the IMS 20. One or both of the IMS 20 and RDBMS 24 can be hosted on a server computer 28, or each can have its own associated computer.

As intended herein, either or both of the user computer 12/server computer 28 can be a server computer made by International Business Machines Corporation (IBM) of Armonk, N.Y. Other digital processors, however, may be used, such as personal computers, laptop computers, mainframe computers, palmtop computers, personal assistants, or any other suitable processing apparatus can be used.

In any case, the processor of the computers access appropriate software to undertake the logic of the present invention, which may be executed by a processor as a series of computer-executable instructions. In the preferred embodiment shown in Figure 1, the software is contained in a software layer or module 30 that is between the IMS 20 and RDBMS 24 in the architecture shown or alternately that is inside the IMS 20, and that is accessed by the IMS 20 and/or RDBMS 24. The instructions contained in the module 30 may be stored on a data storage device with a computer readable medium, such as a computer diskette having a computer usable medium with a program of instructions stored thereon. Or, the instructions may be stored on random access memory (RAM) of a computer, on a DASD array, or on magnetic tape, conventional hard disk drive, electronic read-only memory, optical storage device, or other appropriate data storage device. In an illustrative embodiment of the invention, the computer-executable instructions may be lines of C or C++ or Java code.

Indeed, the flow charts herein illustrate the structure of the logic of the present invention as embodied in computer program software. Those skilled in the art will appreciate that the flow charts illustrate the structures of computer program code elements including logic circuits on an integrated

circuit, that function according to this invention. Manifestly, the invention is practiced in its essential embodiment by a machine component that renders the program code elements in a form that instructs a digital processing apparatus (that is, a computer) to perform a sequence of function steps corresponding to those shown.

5 Commencing at block 32 in Figure 2, in one embodiment a referential integrity (RI) table is created containing a unique identification column. Such a table is shown in Figure 3 and designated 34, containing ID column 46. Multiple RI tables 34 can be used in IMS 20 partitions, if desired.

As shown in Figure 3, an unscoped or heterogeneously scoped link 36 from an RDBMS table 38 ("the referencing tuple") maintains a direct link 41 to a target tuple in a target RDBMS table 42 ("the referenced tuple"). In the present invention, when referential constraint is requested, the link 36 is also related to a row 44 of the RI table 34 by means of a unique identification in the ID column 46. Other rows 48 of the RI table 34 represent other unscoped or heterogeneously scoped links. As indicated at blocks 50 and 52 of Figure 2, for each data table created or used by the IMS 20, a null-able RDBMS foreign key with a RESTRICT delete rule and referencing the corresponding data table, is added to a foreign key column 54 of the RI table 34.

Figure 4 shows that at run time, a link reference (i.e., to a referred-to table) is received based on a user input at block 56. For instance, a user might insert or update a tuple to create an instance of a heterogeneously scoped or unscoped link to a referenced tuple. The scoping constraint, if any, is checked. At decision diamond 58 it is determined whether referential constraint has been requested, and if not the process ends at state 60. Thus, the present invention provides for optionally not maintaining referential integrity for a particular link.

On the other hand, when referential constraint is requested the logic moves to block 62 to insert a row into the RI table 34, containing a unique ID identifying the row and a foreign key pointing to the referred-to tuple. Moving to block 64, all other foreign key fields are set to NULL. Proceeding to block 66, the unique is stored in the link reference to refer to the inserted RI table 34 row. For retrieval and traversal operations involving the link, the direct reference 41 (Figure 3) is used, thereby avoiding the overhead of indirection. To unlink, such as to nullify the link 36 or to delete the referencing tuple 38 from its RDBMS table, the corresponding row 44 of the RI table 34 is also deleted using the ID stored in the link at block 66. To change the target of a link to a different referenced tuple, the steps are the same as those of unlinking followed by those of linking.

To support SET NULL and CASCADE delete rules in addition to the RESTRICT rule, an inverse reference can be added to the RI tables, identifying the source unscoped link instance. Triggers can be defined on the RI tables to change a propagated deletion or update on the RI table to a deletion of the respective RI row, and either an update of the source unscoped link to NULL, or a deletion of the respective referencing tuple.

Now referring to Figures 5-9, an alternate means for ensuring referential integrity for non-homogeneously scoped links can be seen. Commencing at block 80 in Figure 9, an ICMULINK user defined type (UDT) is defined for columns that contain heterogeneously scoped links. In addition, at least one link table is created, as are a Scope table and a Column table.

Specifically, Figure 5 illustrates the details of the ICMULINK UDT. As shown, the UDT uses a fixed-length character string (preferably a CHAR(70) SQL data type) as the base data type, and sufficient information is encoded in the string to undertake the logic below. The module 30 interprets the string as consisting of a number of fields laid out in a predefined format, but this

format is hidden from the application 14, which only uses well-defined user-defined functions (UDFs) to query and operate on the data in the RDBMS 24.

As shown in the particularly preferred ICMULINK UDT shown in Figure 5, a LinkID field is a 16 byte field that is a unique identifier for an unscoped (or heterogeneously-scoped) link. An 18 byte LinkTableName field contains the name of the link table in which the link resides. Also, an 18 byte TargetTableName field contains the name of the target table associated with the link.

Still considering Figure 5, the primary key of the target tuple in the target table is contained in an eight byte TargetID field. The TargetID field contains a 64-bit integer (corresponding to the BIGINT SQL data type). Further, a single byte field represents delete semantics associated with the link. In one preferred embodiment, "1" indicates RESTRICT delete rule, "2" indicates SET NULL, and "3" indicates CASCADE delete. A one-byte field indicating whether referential integrity is to be maintained follows, and then a 4-byte field indicating a user-specified link type. Finally, an unused field is provided for expansion.

In Figure 6, the link table 74 contains one row for each heterogeneously scoped or unscoped link for which referential integrity is to be maintained. The schema-definition of the Link table 74 in SQL is:

```
CREATE TABLE ICMSTLINKTABLE0001(  
    LINKID CHAR(16) NOT NULL,  
    SRCTBLNM CHAR(18),  
    SRCID BIGINT,  
    SRCCOLNM VARCHAR(18),  
    DSTTBLNM CHAR(18),  
    DSTID BIGINT,  
    DELSEM CHAR(1),  
    PRIMARY KEY(LINKID)  
)
```

In the above schema the 16-byte link identifier is the primary key for the table. Columns 2, 3, and 4 together locate the heterogeneously scoped link column, whereas columns 5 and 6 uniquely identify the referenced tuple. To improve the performance of the below-described triggers, the following index is defined on the Link table 74:

```
5 CREATE INDEX ICMDESTIDX ON ICMSTLINKTABLE0001(DSTTBLNM, DSTID);
```

Having described the Link table, Figure 7 shows an exemplary Scope table 76. Essentially, the Scope table 76 groups a collection of tables into a "scope" by correlating a scope ID with the names of the tables in the scope. A table may participate in multiple scopes. The schema-definition in SQL for the Scope Table is:

```
CREATE TABLE ICMSTSCOPEDEFN(  
    SCOPEID INTEGER NOT NULL,  
    TABLENAME VARCHAR(18) NOT NULL  
)  
CREATE INDEX ICMSTSCOPEDEFNINDX ON ICMSTSCOPEDEFN(SCOPEID)
```

Now considering Figure 8, an exemplary Column table 78 can be seen. For each heterogeneously scoped or unscoped link column defined in the database, a row is maintained in the Column table. The Column table 78 contains both the name of the column (ULINKCOLNAME) and name of the table containing the column (TABLENAME), along with the scope ID for the link column and a delete semantics byte that is set according to the above principles. The Column table 78 has the following schema definition:

```
20 CREATE TABLE ICMSTULCOLINFO(  
    TABLENAME VARCHAR(18) NOT NULL,  
    UNLINKCOLNAME VARCHAR(18) NOT NULL,  
    SCOPEID INTEGER NOT NULL,  
25    DELSEM CHAR(1) NOT NULL,  
    PRIMARY KEY(TABLENAME, ULINKCOLNAME)  
)
```

At block 82 in Figure 9, for each table that could potentially be the target of a heterogeneously scoped link, a DO loop is entered in which UPDATE and DELETE triggers are created on the table at block 84. The appropriate one of the triggers is respectively activated only when an update operation changes the primary key of an affected tuple or when a delete operation is executed on one or more tuples. The Link table 74 (and, for column-level operation, the Column table 78) is checked to determine whether a tuple sought to be updated is a target tuple of a heterogeneously-scoped link, and if so the update operation is disallowed. In the case of a delete, the delete trigger checks the Link table to determine whether the tuple is listed as a target tuple of a link with RESTRICT delete semantics associated with it. If so, the delete is disallowed at block 88. Otherwise, the delete is allowed.

Exemplary respective outlines for the update and delete triggers are as follows:

```
CREATE TRIGGER ICMUPDATE000000123
NO CASCADE BEFORE UPDATE OF ID ON T
REFERENCING OLD AS O
FOR EACH ROW MODE DB2SQL
WHEN (EXISTS (SELECT*FROM ICMSTLINKTABLE0001
              WHERE DSTTABLENAME = 'T' AND DSTID = O.ID))
      SIGNAL SQLSTATE '85001' (CANNOT UPDATE TARGET OF UNSCOPED
                              LINK)
```

```
CREATE TRIGGER ICMDELETE000000124
NO CASCADE BEFORE UPDATE OF ID ON T
REFERENCING OLD AS O
FOR EACH ROW MODE DB2SQL
WHEN (EXISTS (SELECT*FROM ICMSTLINKTABLE0001
              WHERE DSTTABLENAME = 'T' AND DSTID = O.ID AND
              DELSEM='1'))
      SIGNAL SQLSTATE '85002' (CANNOT DELETE TARGET OF UNSCOPED
                              LINK)
```

10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
260
265
270
275
280
285
290
295
300
305
310
315
320
325
330
335
340
345
350
355
360
365
370
375
380
385
390
395
400
405
410
415
420
425
430
435
440
445
450
455
460
465
470
475
480
485
490
495
500
505
510
515
520
525
530
535
540
545
550
555
560
565
570
575
580
585
590
595
600
605
610
615
620
625
630
635
640
645
650
655
660
665
670
675
680
685
690
695
700
705
710
715
720
725
730
735
740
745
750
755
760
765
770
775
780
785
790
795
800
805
810
815
820
825
830
835
840
845
850
855
860
865
870
875
880
885
890
895
900
905
910
915
920
925
930
935
940
945
950
955
960
965
970
975
980
985
990
995
1000

The present invention recognizes that information stored in the ICMULINK attribute might be packaged in a proprietary format with internally meaningful identifiers and codes, an application-friendly interface is desirable. These are provided in the preferred embodiment by a set of UDFs that operate on the ICMULINK attributes. A UDF can be provided to parse the internal structure of an ICMULINK attribute to provide applications with individual field values. Also, a UDF can be provided to determine whether two link attributes passed as input point to the same tuple. Moreover, a UDF can be provided to generate ICMULINK attributes according to the above format using, as input, the target and linktype information, as well as the other parameters shown in Figure 5. Additionally, a UDF can be provided to retarget an unscoped link, in the event that an application finds it useful to change the target of a link without changing any of its properties.

At block 86 in Figure 9, for each heterogeneously scoped or unscoped link column defined in the database using the ICMULINK data type, a DO loop is entered in which a row is inserted into the Column table at block 88, and one or more rows are also inserted into the Scope table if a new scope is defined. When applications create or delete tables in the RDBMS 24 and insert or delete or modify tuples according to allowed operations as provided for in Figure 9 above, the module 30 preferably modifies the tables shown in Figures 6-8 accordingly. To do this, the module 30 includes a set of RDBMS-stored procedures that are provided to IMS 20 and applications 14 so that an application can invoke the relevant stored procedure in conjunction with update, delete, and insert operations, as indicated at blocks 90 and 92 in Figure 9. It is to be understood that, as an alternative, the same logic can be implemented natively in the module 30 instead of a RDBMS stored procedures.

When an application creates a new table, it calls a prepareTarget stored procedure, passing it the name of the table being created and the name of the primary key column. The stored

procedure generates and executes the necessary triggers discussed previously at blocks 82 and 84.

In C code, this stored procedure can be represented by:

```
void ICMPrepareTarget(char*tableName, char*pkeyColumnName)
```

When a new link column is added as discussed at blocks 86 and 88, a stored procedure is called with the following input parameters to create a new entry in the Column table 78: name of table containing new column, name of column, scopeID associated with the column (0 by default), and column-level delete semantics to be applied (0 by default).

In C code, this stored procedure can be represented by:

```
void ICMRegisterULinkColumn(char*tableName, char*LinkColumnName, long scopeID, char delSem)
```

For dropping a table, an application is provided with the following stored procedure, using the table name as sole parameter:

```
void ICMDropTable(char*tableName)
```

The above procedure first checks the Link table 74 for links pointing to tuples in the table to be dropped, and if any exist, the table is prevented from being dropped. Otherwise, all system entries pertaining to the table are deleted.

In contrast, to insert a tuple in a table, the name of the table, primary key of the tuple, link being inserted as part of the tuple insertion, and column name of the link are passed to the following stored procedure, which is invoked by an application:

```
void ICMULinkInsert(char*tableName, sqlint64 primaryKey, char*Link, char *LinkColName)
```

The above procedure first determines whether the target table referenced by the link to be inserted is part of the scope that is associated with the link column, using the Scope table 76 and Column table 78. If the check fails the procedure returns an exception. Otherwise, the procedure

next determines whether referential integrity is to be maintained for the new link, and if not returns and ends. When referential integrity is to be maintained, however, the procedure accesses the target table to determine whether the tuple referenced by the new link exists, and if not an exception is returned, but if the tuple is found the procedure creates a new entry in the Link table 74.

5 For deleting a tuple containing a link attribute, a procedure is called to determine whether referential integrity is to be maintained for the link, and if so the corresponding entry in the Link table 74 is deleted. The procedure is called by an application using the link attribute being deleted as input, and the procedure can accordingly be represented as void ICMULinkDelete(char*link).

10 When an application wishes to update a link attribute of a tuple, it passes the name of the relevant table, primary key of the tuple, old and new values of the link attribute, and name of the link value being modified to the following stored procedure:

void ICMULinkModify(char*tableName, sqlint64 primaryKey, char* OldLink, char* NewLink, char* colName).

15 The procedure combines the above operations of the delete and insert procedures to undertake the update.

To update the primary key of a tuple containing a link attribute, an application sends the new primary key value and link attribute of the tuple to a procedure which, when referential integrity is being maintained, extracts the link ID and locates and modifies the corresponding entry in the Link table 74. The procedure can be represented as:

20 void ICMPkeyModify(char* primaryKey, char* uLink)

10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
260
265
270
275
280
285
290
295
300
305
310
315
320
325
330
335
340
345
350
355
360
365
370
375
380
385
390
395
400
405
410
415
420
425
430
435
440
445
450
455
460
465
470
475
480
485
490
495
500
505
510
515
520
525
530
535
540
545
550
555
560
565
570
575
580
585
590
595
600
605
610
615
620
625
630
635
640
645
650
655
660
665
670
675
680
685
690
695
700
705
710
715
720
725
730
735
740
745
750
755
760
765
770
775
780
785
790
795
800
805
810
815
820
825
830
835
840
845
850
855
860
865
870
875
880
885
890
895
900
905
910
915
920
925
930
935
940
945
950
955
960
965
970
975
980
985
990
995
1000

The above stored procedure first handles the change to the link attribute by invoking the ICMULinkModify stored procedure, passing the "old" primary key to support this update, and then the new primary key is passed to complete the procedure.

Completing the description of the stored procedures, an expected common operation in digital libraries is the migration of a tuple from one table to another. Accordingly, an application wishing to do this can invoke an ICMigrateTarget stored procedure by passing to it the old and new table names and primary key of the tuple. The procedure first determines, from the Link table 74, whether a link exists having the tuple as a target, and for each such link the new location of the tuple and corresponding link attributes of source tuples are changed in the Link table. This procedure, in C code, can be expressed as `void ICMigrateTarget(char*origTable, char*newTable, sqlint64 primaryKey).`

Preferably, an application using the above procedures ensures that the SQL statement and procedure calls are executed together in one atomic statement, so that if the call to the stored procedure does not succeed, the SQL statement can be rolled back.

While the particular SYSTEM AND METHOD FOR ENSURING REFERENTIAL INTEGRITY FOR HETEROGENEOUSLY SCOPED REFERENCES IN AN INFORMATION MANAGEMENT SYSTEM as herein shown and described in detail is fully capable of attaining the above-described objects of the invention, it is to be understood that it is the presently preferred embodiment of the present invention and is thus representative of the subject matter which is broadly contemplated by the present invention, that the scope of the present invention fully encompasses other embodiments which may become obvious to those skilled in the art, and that the scope of the present invention is accordingly to be limited by nothing other than the appended claims, in which reference

to an element in the singular is not intended to mean "one and only one" unless explicitly so stated, but rather "one or more". All structural and functional equivalents to the elements of the above-described preferred embodiment that are known or later come to be known to those of ordinary skill in the art are expressly incorporated herein by reference and are intended to be encompassed by the present claims. Moreover, it is not necessary for a device or method to address each and every problem sought to be solved by the present invention, for it to be encompassed by the present claims. Furthermore, no element, component, or method step in the present disclosure is intended to be dedicated to the public regardless of whether the element, component, or method step is explicitly recited in the claims. No claim element herein is to be construed under the provisions of 35 U.S.C. §112, sixth paragraph, unless the element is expressly recited using the phrase "means for" or, in the case of a method claim, the element is recited as a "step" instead of an "act".

WE CLAIM:

CLAIMS

1 1. A system for providing referential integrity for heterogenous links, comprising:
2 an RDBMS providing referential integrity for homogenous links; and
3 a software layer on top of the RDBMS for causing the RDBMS to provide referential
4 integrity for heterogenous links.

1 2. The system of Claim 1, wherein the software layer maintains at least one table.

3 3. The system of Claim 2, wherein the table is accessed upon an attempted deletion or
updating of a tuple referenced by a link, and the attempted deletion or updating is selectively
disallowed based on the table.

4 4. The system of Claim 2, wherein the software layer includes at least one stored
procedure accessible by an application to insert, update, or delete a tuple while ensuring referential
integrity in heterogenous links associated with the tuple.

1 5. A computer-implemented method for preventing dangling pointers in heterogeneously
2 scoped links, comprising the acts of:
3 providing at least one heterogeneously scoped link (HSL) table, at least one table
4 having a heterogeneously scoped link column, the HSL table being associated with the
5 heterogeneously scoped link column; and

6 accessing the HSL table to ensure referential integrity in an RDBMS.

1 6. The method of Claim 5, wherein the HSL table is accessed when a link attribute is
2 sought to be changed.

1 7. The method of Claim 5, wherein the HSL table is accessed when a tuple is sought to
2 be changed or deleted.

1 8. The method of Claim 5, wherein the HSL table is established by an RI table.

9. The method of Claim 5, further comprising providing at least one trigger useful in
selectively disallowing operations.

10. The method of Claim 5, further comprising providing at least one stored procedure
accessible by an application to insert, update, or delete a tuple while ensuring referential integrity in
heterogenous links associated with the tuple.

1 11. A computer program product including computer usable code means programmed with
2 logic for ensuring referential integrity in an RDBMS having at least one table with at least
3 one column of heterogeneously scoped links, the program product comprising:
4 computer readable code means for maintaining a table; and

5 computer readable code means for using the table to ensure that operations on tuples
6 do not result in a heterogeneously scoped link pointing to no tuple.

1 12. The computer program product of Claim 11, further comprising:

2 computer readable code means for establishing at least one trigger useful in
3 cooperation with the table for selectively disallowing operations.

1 13. The computer program product of Claim 12, further comprising computer readable
2 code means for establishing at least one of: a delete trigger, and an update trigger.

14. The computer program product of Claim 11, further comprising computer readable
code means for inserting, updating, or deleting a tuple while ensuring referential integrity in
heterogenous links associated with the tuple.

18. A system for supporting trigger, comprising:

an RDBMS providing referential integrity for homogenous links; and

a software layer on top of the RDBMS for causing the RDBMS to support triggers.

1 19. The system of Claim 18, wherein the software layer maintains at least one table and
2 establishes at least one trigger.

1 20. The system of Claim 19, wherein the software layer establishes at least one of: a delete
2 trigger, and an update trigger.

1 21. The system of Claim 20, wherein the trigger accesses the table upon an attempted
2 deletion or updating of a tuple referenced by a link, and the attempted deletion or updating is
3 selectively disallowed based on the table.

1 22. The system of Claim 19, wherein the software layer includes at least one stored
2 procedure accessible by an application to insert, update, or delete a tuple while ensuring referential
3 integrity in heterogenous links associated with the tuple.

**SYSTEM AND METHOD FOR ENSURING REFERENTIAL INTEGRITY
FOR HETEROGENOUS REFERENCES IN AN INFORMATION
MANAGEMENT SYSTEM**

ABSTRACT OF THE DISCLOSURE

5 A system and method are provided for an information management system (IMS) to manage heterogenous references in the system, to ensure "referential integrity", without changing the underlying relational database management system (RDBMS) of the IMS. One or more system tables are kept that are not visible to system users. In one embodiment, the RDBMS' mechanisms to ensure referential integrity for homogenous references is used in conjunction with the system table to extend referential integrity to heterogenous references. In another embodiment, the triggers of the RDBMS, in conjunction with the system table, are used to ensure referential integrity of heterogenous references.

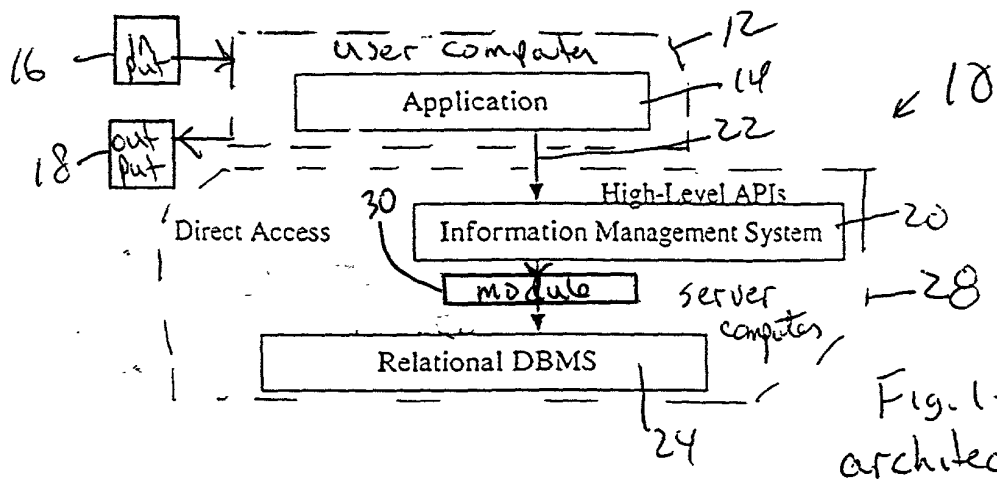


Fig. 1-
architecture

002720" 63E4F360

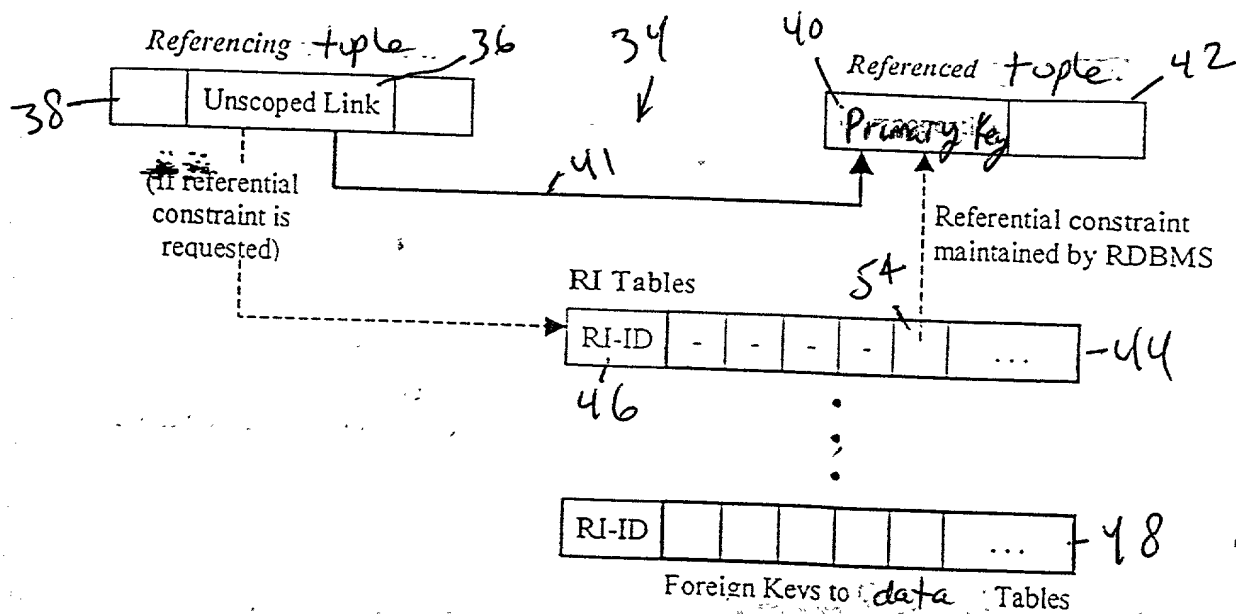
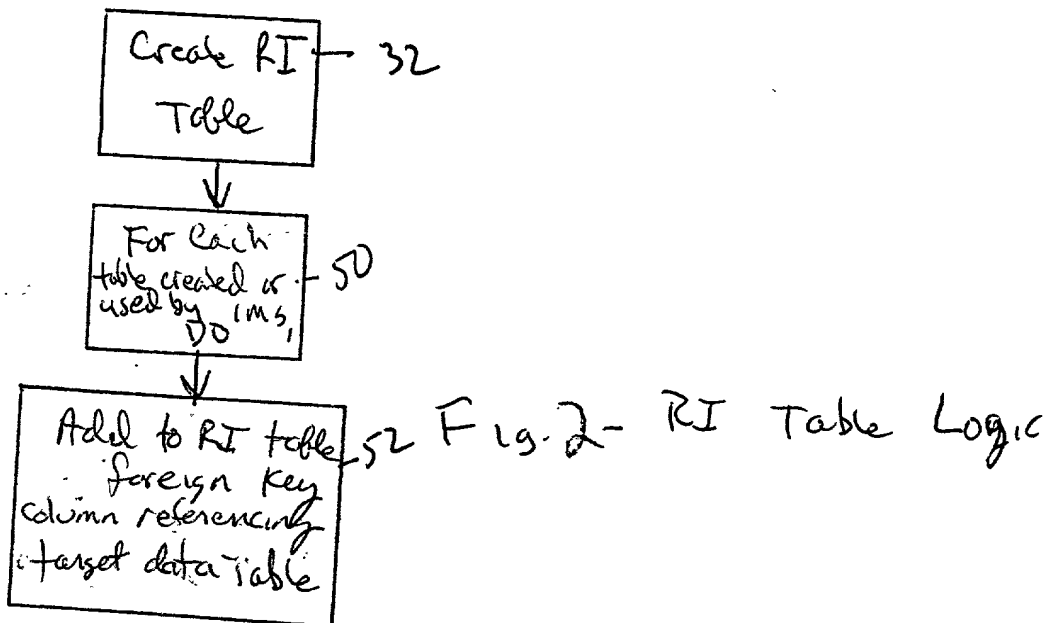
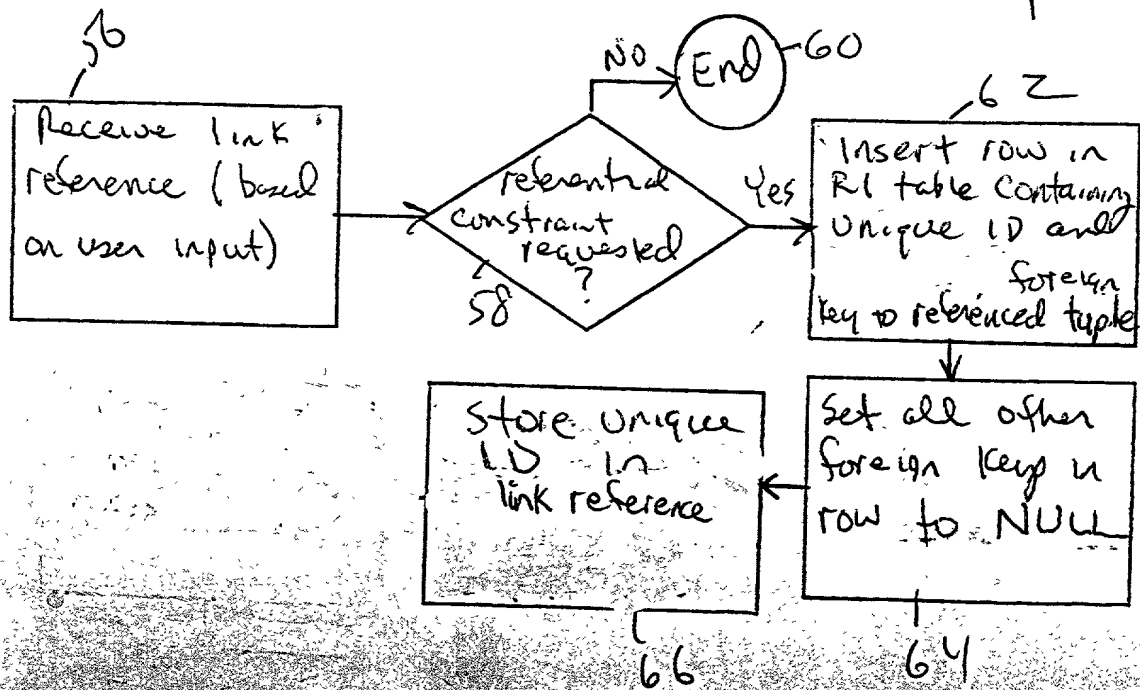


Fig. 3 RI Table

Fig. 4 - RI Table operation



Name of the Field	Field Size	Byte Numbers	Description
LinkID	16	0 to 15	Unique identifier for the unscoped link
LinkTableName	18	16 to 33	Name of the linktable in which entry for this link will reside
TargetTableName	18	34 to 51	Name of the target table
TargetID	8	52 to 59	Primary key of the target tuple
DeleteSemantics	1	60	'1' indicates RESTRICT '2' indicates SET NULL '3' indicates CASCADE
NeedRIIndicator	1	61	'1' if referential integrity needs to be maintained for this link. '0' otherwise
LinkType	4	62 to 65	User-specified integer
Unused	4	66 to 69	For future use and expansion

Fig. 5

Fig. 7

SCOPEID	TABLERNAME
10	THESES
10	CONFPAPERS
10	SURVEYS
10	TECHREPS
10	JOURNALPAPERS

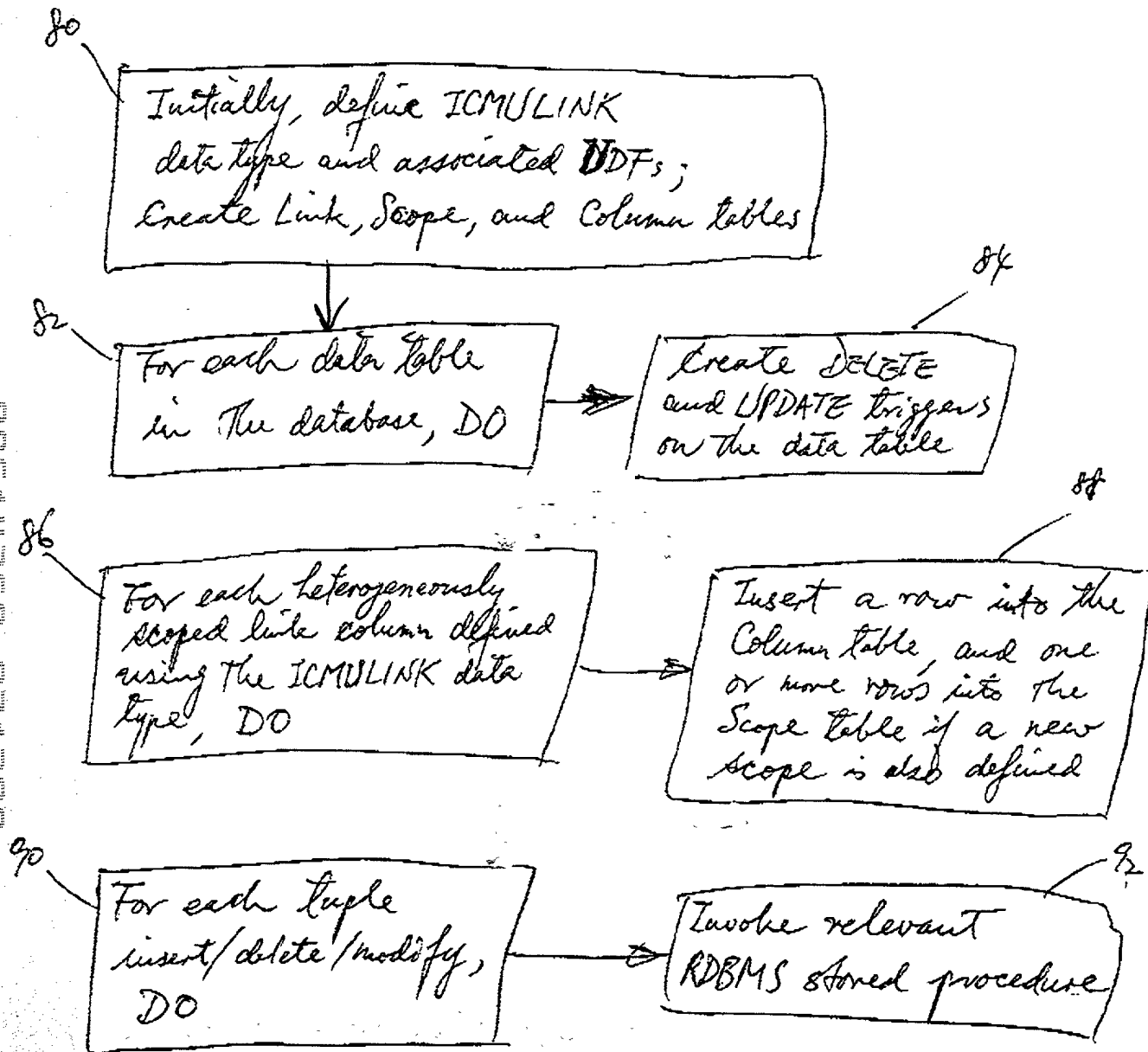
Fig. 8

TABLERNAME	ULINKCOLNAME	SCOPEID	DELSEM
STUDENTPUBS	PUB	10	'1'
FACULTYPUBS	PUB	10	'1'

Fig. 6

Link ID	Source Table Name	Source ID	Source Column name	Dest. Table Name	Dest. Table ID	DEL SEM
---------	-------------------	-----------	--------------------	------------------	----------------	---------

Fig 9. - alternate logic



DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

SYSTEM AND METHOD FOR ENSURING REFERENTIAL INTEGRITY FOR HETEROGENEOUSLY SCOPED REFERENCES IN AN INFORMATION MANAGEMENT SYSTEM

the specification of which is attached hereto unless the following box is checked:

was filed on _____
as United States Application Number or PCT International Application Number _____
and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR §1.56.

I hereby claim foreign priority benefits under 35 USC §119(a-d) or §365(b) of any foreign application(s) for patent or inventor's certificate, or §365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below, by checking the box, any foreign application for patent or inventor's certificate, or PCT International application having a filing date before that of the application on which priority is claimed.

Prior Foreign Application(s):**Priority Not Claimed**

_____ (Number)	_____ (Country)	_____ (Day/Month/Year Filed)
-------------------	--------------------	---------------------------------

I hereby claim the benefit under 35 USC §119(e) of any United States provisional application(s) listed below:

Provisional Application(s):

_____ (Application Number)	_____ (Filing Date)
-------------------------------	------------------------

I hereby claim the benefit under 35 USC §120 of any United States application(s), or §365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of 35 USC §112, I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR §1.56 which became available between the filing date of the prior application and the national or PCT International filing date of this application.

_____ (Application Number)	_____ (Filing Date)	_____ (Status - patented, pending, abandoned)
-------------------------------	------------------------	--

Power of Attorney:

I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:

Richard M. Ludwin	(#33,010)	Khanh Q. Tran	(#41,352)
Thomas R. Berthold	(#28,689)	Alison D. Mortinger	(#39,306)
Marc D. McSwain	(#44,929)	John L. Rogitz	(#33,549)

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

Address all telephone calls to:

John L. Rogitz

(619) 338-8075

Address all correspondence to:

John L. Rogitz

Rogitz & Associates

750 B Street, Suite 3120

San Diego, California 92101

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor: **DAVID MUN-HIEN CHOY**

Inventor's signature:

Date:

David Choy
*June 22, 2000*Residence: **1300 Bright Oaks Court, Los Altos, California 94024**Citizenship: **United States of America**

Post Office Address: Same

Full name of second inventor: **SRIRAM RAGHAVAN**

Inventor's signature:

Date:

Residence: **3075 Silver Estates, San Jose, California 95135**Citizenship: **India**

Post Office Address: Same

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

SYSTEM AND METHOD FOR ENSURING REFERENTIAL INTEGRITY FOR HETEROGENEOUSLY SCOPED REFERENCES IN AN INFORMATION MANAGEMENT SYSTEM

the specification of which is attached hereto unless the following box is checked:

was filed on _____
 as United States Application Number or PCT International Application Number _____
 and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR §1.56.

I hereby claim foreign priority benefits under 35 USC §119(a-d) or §365(b) of any foreign application(s) for patent or inventor's certificate, or §365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below, by checking the box, any foreign application for patent or inventor's certificate, or PCT International application having a filing date before that of the application on which priority is claimed.

Prior Foreign Application(s):

Priority Not Claimed

_____ (Number)	_____ (Country)	_____ (Day/Month/Year Filed)
-------------------	--------------------	---------------------------------

I hereby claim the benefit under 35 USC §119(e) of any United States provisional application(s) listed below:

Provisional Application(s):

_____ (Application Number)	_____ (Filing Date)
-------------------------------	------------------------

I hereby claim the benefit under 35 USC §120 of any United States application(s), or §365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of 35 USC §112, I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR §1.56 which became available between the filing date of the prior application and the national or PCT International filing date of this application.

_____ (Application Number)	_____ (Filing Date)	_____ (Status - patented, pending, abandoned)
-------------------------------	------------------------	--

Power of Attorney:

I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:

Richard M. Ludwin	(#33,010)	Khanh Q. Tran	(#41,352)
Thomas R. Berthold	(#28,689)	Alison D. Mortinger	(#39,306)
Marc D. McSwain	(#44,929)	John L. Rogitz	(#33,549)

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

Address all telephone calls to:

John L. Rogitz

(619) 338-8075

Address all correspondence to:

John L. Rogitz

Rogitz & Associates

750 B Street, Suite 3120

San Diego, California 92101

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor: **DAVID MUN-HIEN CHOY**

Inventor's signature:

Date:

Residence: **1300 Bright Oaks Court, Los Altos, California 94024**Citizenship: **United States of America**

Post Office Address: Same

Full name of second inventor: **SRIRAM RAGHAVAN**

Inventor's signature:

Date:

6/27/2000Residence: **3075 Silver Estates, San Jose, California 95135**Citizenship: **India**

Post Office Address: Same